An approach towards decentralized control of cooperating non-autonomous multiple robots

Robotica (2000) volume 18, pp. 495–504. Printed in the United Kingdom © 2000 Cambridge University Press

An approach towards decentralized control of cooperating non-autonomous multiple robots

Khalid Munawar*, Masayoshi Esashi† and Masaru Uchiyama*

(Received in Final Form: November 2, 1999)

SUMMARY

This paper introduces an event-based decentralized control scheme for the cooperation between multiple manipulators. This is in contrast to the common practice of using only centralized controls for such cooperation which, consequently, greatly limit the flexibility of robotic systems. The manipulators used in the present system are very simple with only two degrees of freedom, while even one of them is passive. Moreover these manipulators use very few and commonly available sensors only. Computer simulations indicated the applicability of the event-based decentralized control scheme for multi-manipulator cooperation, while real-life experimental implementation has proved that the proposed decentralized control scheme is fairly applicable for very simple and even under-actuated systems too. Hence, this work has opened new doors towards further research in this area. The proposed control scheme is expected to be equally applicable for any mobile or immobile multi-robotic system.

KEYWORDS: Decentralized control; Multiple robots; Cooperation.

1. INTRODUCTION

The approach of multiple manipulators' cooperation is closer to human behavior and guarantees a better stability than is realized with a single manipulator. Moreover it increases the dexterity of a system enabling it to realize more complicated and intelligent jobs with better degree of skill. Besides this, it also becomes possible to avoid some unwanted states, like singularities etc., during execution of jobs.

In the past, many people have studied dual manipulator systems. Many different cooperation strategies and control approaches corresponding to these strategies have been introduced.[1] However, almost all of these cooperation schemes are extensions of classical centralized control approaches. Therefore, huge computational powers are essentially required to implement these centralized dual manipulator cooperation schemes.

With the modern developments and demands, the systems are no more limited to two manipulators only, but people are now considering multiple manipulators in a system. The most common examples of such systems are found in automated assembly lines and micro-level manipulations[2–4] etc. For such huge and complex systems, it has become almost impractical to use centralized controllers because of limitations in the computational powers and hardware implementations. Moreover, even if centralized controllers are developed, they cannot guarantee a further expansion or flexibility in the systems. Besides this, the approach of a centralized controller is against the natural systems also where multiple agents cooperate but without any centralized command and control. Therefore the approach of Distributed Autonomous Control has been thought very attractive and warmly welcomed since its introduction.[4] As this approach was inspired by the natural systems and cooperation between them, hence much better results could be expected. Many people have worked intensively on this approach and various applications and developments are reported.[2–6] Normally, the distributed control approaches applied to robotic system assume that an individual robot in the distributed environment is fully autonomous in sensing and in executing jobs. Therefore, until now, we can find only autonomous mobile robots or very intelligent manipulators utilizing this approach, while examples of semi-autonomous or non-autonomous systems working decentralized are rare.

Another novel and very interesting approach which has enhanced the attraction of distributed control is the event-based control introduced in reference [7]. This control approach has made the systems task or event dependent instead of on time, and hence it is possible now for any general robotic system, particularly a distributed system, to execute the assigned tasks with greater stability. Especially, when there arise some faults or robot motion is hampered by some external unknown object, this approach guarantees both stability of control and safety of robot and its environment. In this sense, it is now easier to work in hazardous areas. However, the event-based control approach has also found its application for very intelligent and dextrous robotic system with a high degree of autonomy.

This paper has targeted the application of both distributed and event-based control approaches for non-autonomous systems. Consequently, a control strategy is developed for cooperation in a decentralized and immobile multiple manipulator system. This system has practically non-autonomous robots with only two degrees of freedom, while only one of them is active, and they utilize minimal sensory information. An event-based supervisory control scheme is proposed to exhibit a stable distributive behavior. However, the supervisor does nothing but only manages the communication of a very small data among the individual robots. In this sense, it only acts as a lookup table from where the

* Graduate School of Engineering, Tohoku University
† New Industry Creation Hatchery Center, Tohoku University, Aoba-yama 01, Aoba-ku, Sendai 980-8579 (JAPAN)

individual robots can access the command parameters and then write back their task execution results.

As a first step, different computer simulations are performed which have shown the practicability of the proposed control approach for even very poor systems like the one considered here.[8] Moreover, these simulations have given idea about different control parameters which could have become a big problem during experimental implementations. As an ultimate target, the proposed control strategies have been experimentally implemented for a real-life multiple manipulator system. The very encouraging results have proved the applicability of distributed and event-based approaches, in the proposed manner, for a practically poor system. This achievement has opened doors for further research and has made it possible to think about developing bigger systems with very simple but close to autonomous functional elements.[9]

2. THE SYSTEM UNDER CONSIDERATION

Figure 1(a) shows an overview of the experimental system being considered in this paper for development of the ideas. The main theme behind the development of this system is to simulate a cooperative micro-manipulation system. The structure of a 2 d.o.f implementation of the micro-actuator which gave arise to the development of its macro-level simulator, being discussed in this paper, is shown in Figure 2(a). It is a mechanism with four micro actuator units to develop both positive and negative motions about a mean position, and an active/passive second d.o.f. An actuator unit is a cascaded structure of many electrostatic functional elements. The structure of such an electrostatic element and its operation is explained in Figure 2(b).[10,11] As only compression can be realized with such actuators, hence two actuator units are cascaded to achieve a desired motion at an end-effector formed between the two, as is the case of Figure 2(a). Another development of the same electrostatic elements in a bellow-shaped actuator structure is shown in Figure 2(c).[12]

Considering the main requirement of simplicity in the micro-level systems, the present macro-level system also realizes very simple robots, named “Manipulation Elements (MEs)”, with only two degrees of freedom, while one of them is kept essentially passive (Figure 1(b)). The motion of the passive joint is, however, constrained by a spring-damper assembly to an equilibrium position. Hence, a lateral force is developed depending upon the angle of the rotary joint. A resultant of active and passive forces is

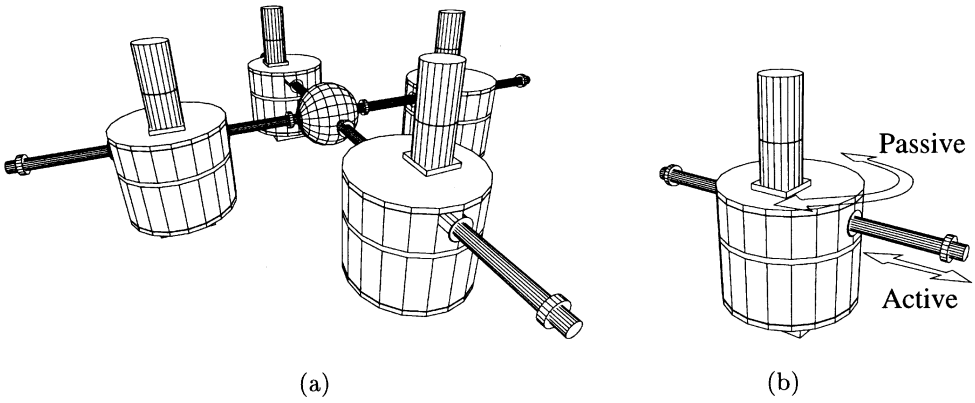

Fig. 1. Cooperative manipulation system with unifunctional robots.

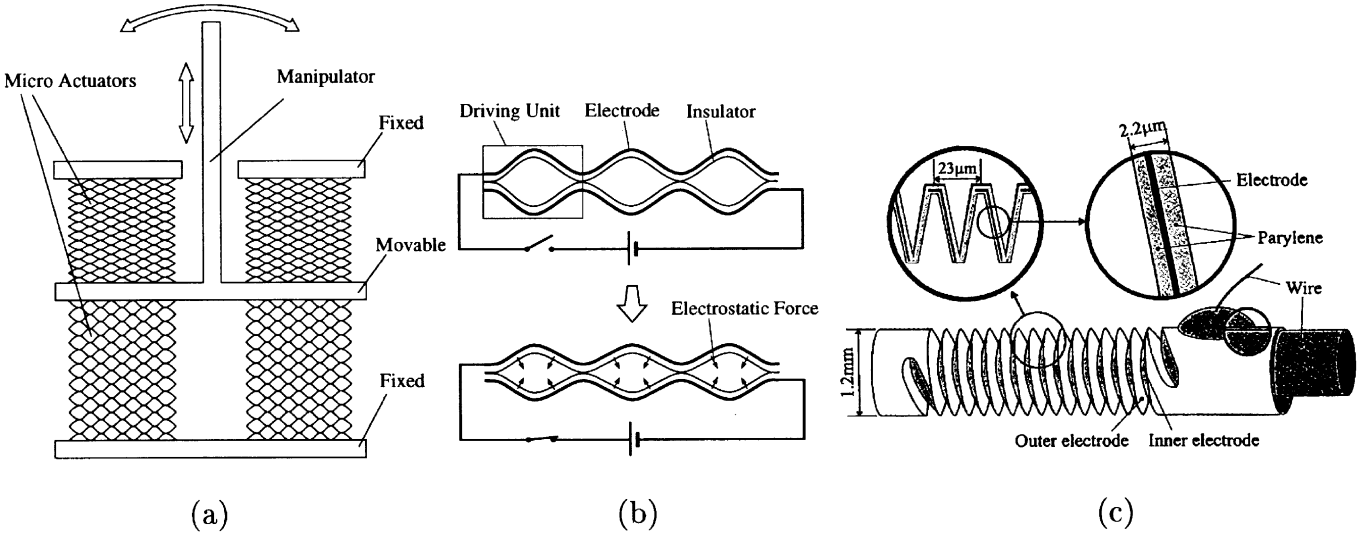

Fig. 2. Electrostatic micro actuator, its operation and a new development.

available at the finger-tip when it is constrained against some external environment, e.g. and object. Therefore, the passive joint can only be utilized when the MEs work in a closed chain.

As this system is intended to manipulate an object in the horizontal plane above the ground, with no vertical support available to the object, hence consideration of the object grasping strategies becomes very important. Normally such systems rely on a centralized control architecture for determination of forces to develop a complete and stable grasp on the object and manipulate it. There are almost no examples of decentralized control methods being used for such grasped manipulation applications.

During execution of such grasps with finger tips only, it is very easy to observe a contact-point slip. However, a controller can never realize these slips and hence the control is lost. It is possible to detect the occurrence of such slips if some dedicated sensors are employed in the system. However, it is not possible to have such sensors in a micro-manipulation system. Therefore, a slip detection method which does not need such dedicated sensors is of vital importance. Previous work on the development of a method for sensorless detection of such contact-point slips can be found in reference [13]. This method can work for any general multiple manipulator system with three or more contact points, such as multi-finger hands or any multi-agent robotic systems.

2.1. Forces generated by a Manipulation Element (ME)

Figure 3 shows the model of an *i*-th ME in contact with an object (not shown). As the considered motions are very small, as is the case in micro systems, hence the dynamics of the system can be ignored. The end-effector/finger force $\boldsymbol{f}_{fi}$ can be simply given as

$$\boldsymbol{f}_{fi} = [F_{mi} \quad F_{si}]^T \tag{1}$$

where F_{mi} is the motor force in the active joint and F_{si} is the force due to spring in the passive joint. The forces along *x*-axis and *y*-axis of a universal frame Σ_o are then known to be

$${}^oF_{xi} = F_{mi}\cos\phi_i - F_{si}\sin\phi_i \tag{2}$$

$${}^oF_{yi} = F_{mi}\sin\phi_i + F_{si}\cos\phi_i \tag{3}$$

where $\phi_i = \theta_{i0} + \theta_i$ is the angle of the push rod with respect to the universal reference frame, while θ_i is the absolute angle (in robot frame) and θ_{i0} is the rotation of the robot frame in the world.

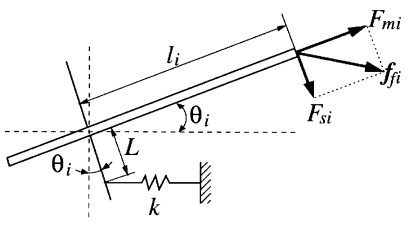

Fig. 3. Static model of a Manipulation Element (ME).

2.2. Centralized control

For a comparison, a conventional centralized control approach is mentioned here, so that is may be possible to clearly notice the benefits of the decentralized approach introduced in this report.

Using the above-given model describing the forces available by an ME, the resulting forces and moment due to *n* number of MEs on an object (identified by Σ_a), corresponding to the three possible motions in a plane, can be easily known as

$$\boldsymbol{F}_a = \sum_{i=1}^{n} {}^o\boldsymbol{f}_{fi} = \left[\sum_{i=1}^{n} {}^oF_{xi} \quad \sum_{i=1}^{n} {}^oF_{yi} \right]^T \tag{4}$$

$$N_{za} = \sum_{i=1}^{n} {}^o\boldsymbol{p}_{fi}^a \times {}^o\boldsymbol{f}_{fi} \tag{5}$$

where ${}^o\boldsymbol{p}_{fi}^a = [{}^ox_{fi}^a \quad {}^oy_{fi}^a]^T$ is the position vector of the finger-tip from the object frame Σ_a as seen by the world frame Σ_o. The above equations can be written in a well-known compact form as

$${}^o\boldsymbol{f}_a = \boldsymbol{W}_m\boldsymbol{q}_m + \boldsymbol{W}_s\boldsymbol{q}_s \tag{6}$$

where $\boldsymbol{W}_m$ is the transformation matrix for the motor forces and $\boldsymbol{W}_s$ is the one for the spring forces of the passive joints, while $\boldsymbol{q}_m$ and $\boldsymbol{q}_s$ are, respectively, the vectors of the motor forces F_{mi} and the spring forces F_{si}. Hence the motor forces can be easily found as

$$\boldsymbol{q}_m = \boldsymbol{W}_m^+({}^o\boldsymbol{f}_a - \boldsymbol{W}_s\boldsymbol{q}_s) + \boldsymbol{N}\boldsymbol{f}_r \tag{7}$$

where $\boldsymbol{N}\boldsymbol{f}_r$ defines the internal or grasping forces on the object, $\boldsymbol{N}$ is the matrix of the null space vectors of $\boldsymbol{W}_m$, and $\boldsymbol{f}_r$ is the magnitude vector of grasping forces. If the dimension of $\boldsymbol{W}_m$ is $m \times n$ such that $m < n$, $\boldsymbol{W}_m^+$ defines the Moore-Penrose psuedo inverse of $\boldsymbol{W}_m$.

The above formulations show that it is very important to know the number of MEs cooperating for a particular task and all the formulations will change if this number changes. Therefore if many MEs are going to execute a task, it will become very difficult and time consuming to redefine complete control algorithm and compute the actuator torques.

3. MODELING OF A MANIPULATION ELEMENT FOR DISTRIBUTED CONTROL

Now as the basic mathematical model for a manipulation element is available, it is possible to go on with the conceptual developments for the decentralized or distributed control. The most important point to note is that, for a robot to work in a distributed environment, it should only consider the status of the ultimate task and rely upon its own sensors to estimate the future behavior. This defines an autonomous behavior of a robot. Therefore, it is desired that a robot should be modeled in contact with the object only, without considering any other agents or robots cooperating for the common task. Moreover, it is also desired that all of a robot's motions be controlled depending upon the motions

of the object only, i.e. that status of the ultimate task. In this section, both of these most important and, perhaps, sufficient steps are discussed.

3.1. Forces on an ME

Consider n number of MEs cooperating to the motions of an object as shown in Figure 4. As only the point contacts with the object are being considered, hence no local moments will be generated at the finger-tips. Therefore, MEs need only to apply forces in the plane but no moments. However, corresponding to the motions of an object in a plane, there must exist a moment around the vertical axis of the object. This desired moment is developed by the collective attitude of all the forces, and hence the object can exhibit a rotation along with two linear motions. This is because, the moments are developed in a constrained object by the application of linear forces. From this observation, it is clear that the control problem, for a single robot, is reduced to the determination of only two variables instead of three, and this happens only if the system is controlled in a distributed fashion. Now, it is not too difficult to determine the only two desired forces, required at a finger-tip, to generate a resultant set of motions when an effective constraint condition can be known by some sensor(s) or method(s). Moreover, it is easy enough to do so with nominal computational resources which can be normally imagined with a single robot.

The finger-tip forces for each ME are known as given in equations (1) through (3). Consider that the i-th ME is fixed in position, while all others are free to exert their forces, then the forces on its finger-tip will be simply the resultant of the forces applied by all other MEs, i.e.

$$\boldsymbol{F}_{fi} = \left[\sum_{k=1}^{n} {}^{o}F_{xk} \quad \sum_{k=1}^{n} {}^{o}F_{yk} \right]^{T}; \quad k \neq i. \tag{8}$$

These will be the forces sensed by a force sensor if one is installed at the finger-tip.

3.2. Finger-tip trajectory prediction

As a second condition for an ME to work independently, it should also predict its own finger-tip trajectories. This is also important for the reason that an ME can compute only two variables, i.e. its own finger-tip force components F_x and F_y, while three variables are desired corresponding to the three object motions. Therefore, it is desired to restrict the motions of an ME to its own controller instead of the object motions. However, an ME needs to predict its trajectories considering the trajectories of the object being manipulated.

Consider again an ME in contact with the object as seen by the reference world frame, shown in Figure 5. As the motions of the object are also considered in the world frame, hence the finger-tip trajectories can be found easily, knowing the position of the finger-tip with respect to the object frame, initially. This is true when there is no slip at the contact-points.

Let ${}^{o}\boldsymbol{p}_{a} = [{}^{o}x_{a} \quad {}^{o}y_{a}]^{\mathrm{T}}$ be the position of the center of mass of the object and ψ_a be the rotation of the object frame Σ_a with respect to the world frame, i.e. object's orientation. Then the finger-tip trajectory of the i-th ME can be found as

$${}^{o}\boldsymbol{p}_{fi} = {}^{o}\boldsymbol{p}_{a} + {}^{o}\boldsymbol{R}_{a}(\psi_{a}) {}^{a}\boldsymbol{l}_{fi} \tag{9}$$

where ${}^{o}\boldsymbol{R}_{a}(\psi_{a})$ is the rotation transformation matrix of Σ_a through an angle ψ_a w.r.t. Σ_o, and ${}^{a}\boldsymbol{l}_{fi}$ is the position vector of the i-th finger-tip with respect to the object frame Σ_a, and it remains unchanged if there is no contact-point slip. Hence corresponding to a reference object posture $[{}^{o}x_{a} \quad {}^{o}y_{a} \quad \psi_{a}]^{T}$, an ME can predict its own reference finger-tip trajectory ${}^{o}\boldsymbol{p}_{fi}$. The individual trajectories of all MEs will then define a resultant set of trajectories same as the desired object trajectories.

4. DISTRIBUTED EVENT-BASED CONTROL

Now, as the important tools for execution of a task are available, it is desired for an ME to generate the proper set of forces as required to take part in the proper manipulation of the object, in cooperation with the other MEs in the system. There are three different types of force components which sum up to the ultimate set of finger-tip forces. Basically, these different types of forces are desired only as the system needs to grasp the object before its manipulation. Moreover, this grasp needs to remain stable throughout the task execution. This section gives an outline of these different forces, the ultimate desired set of forces for an ME, and the controller which can control the whole system in a distributed fashion.

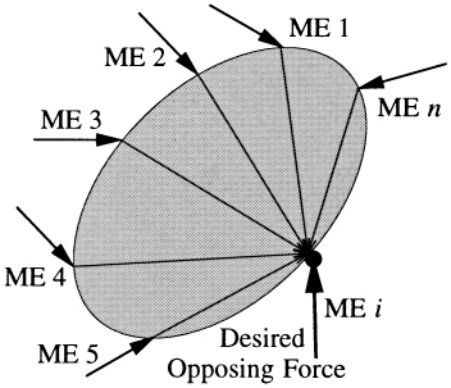

Fig. 4. Forces on an ME in contact with the object.

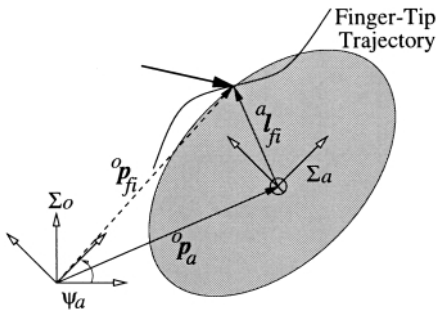

Fig. 5. Determination of an ME's finger-tip trajectory.

4.1. Forces to balance the object

In the start of task execution, it is not known whether the forces exerted by all MEs, on the object, will sum up to a zero force vector or not, so that the object may not exhibit undesired motions. Therefore, it is needed that the MEs should correct their forces as soon as their finger-tips land at the surface of the object. For this purpose, MEs need to have some force sensors installed in their finger-tips, or they should share the current data among themselves to let the other partners know about their current status. Once this data in in hand, it is very easy for an ME to correct its forces to stop the undesired motions of the object due to the forces applied by all other MEs, given in equation (8). Therefore, for an *i*-th ME, it should apply exactly similar but opposite forces to achieve this goal, i.e.

$$\boldsymbol{f}_{i,hold} = -\boldsymbol{F}_{fi}. \tag{10}$$

4.2. Grasping forces developed by trajectory modification

The finger-tip trajectories defined in equation (9) can only define the finger-tip position at the surface of the object. It is not possible for an ME to exert the forces on the object required to grasp it. This is because, it is very difficult for an ME to predict the amount of desired forces for this purpose in the very start of task execution. Therefore, once a force balance is achieved as defined by equation (10), an adjustment is desired in the magnitudes of the finger-tip forces to realize a proper grasp on the object. Moreover, the same adjustment will be needed again even during the task execution when the grasping forces drift a lot causing either a loss of grasp or damaging the object.

A realistic approach is to modify the finger-tip trajectories, as illustrated in Figure 6, such that an ME can exert sufficient forces depending on the stiffness of the object. For this purpose a "**grasping factor** (k_{grasp})" is defined which alters the dimensions of the finger-tip position vector ${}^a\boldsymbol{l}_{fi}$ in the object frame, such that

$${}^a\boldsymbol{l}'_{fi} = (1 - k_{grasp})\,{}^a\boldsymbol{l}_{fi}. \tag{11}$$

This grasping factor can be a constant related to an initial desired magnitude of the internal/grasping forces and the stiffness of the object. It can also be controlled depending upon variations in internal forces. ${}^a\boldsymbol{l}_{fi}$ is specially selected for this purpose as it always links a finger-tip to the center of mass of the object. Hence any imbalance in the forces will at the most change the magnitude of the grasping forces but there will be a little chance of instability of the system or loss of grasp. The forces exerted by an ME on an object of stiffness ρ_a which will contribute in developing a grasp can be stated as

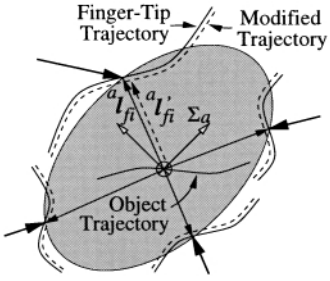

Fig. 6. Trajectory modification for grasping.

$$\boldsymbol{f}_{i,grasp} = \rho_a({}^a\boldsymbol{l}_{fi} - {}^a\boldsymbol{l}'_{fi}). \tag{12}$$

Again, the modified trajectories will also define an object trajectory same as the one defined by original trajectories. However, a component for adjustment of the grasping forces on the object will be added.

4.3. Forces for motions of the object

Until now, all the forces computed are not really adding to the actual desired task, i.e. the manipulation of the object. Therefore, finally, a force component is required which can generate the desired motions in the object. This force can be simply computed by multiplying the finger-tip trajectory error $\Delta^o\boldsymbol{p}_{fi}$ with a control gains matrix $\boldsymbol{K}_{fi}$ having the components of gain for both *x* and *y* motions of a finger-tip, and it can be written as

$$\boldsymbol{f}_{i,motion} = \boldsymbol{K}_{fi}\,\Delta^o\boldsymbol{p}_{fi}. \tag{13}$$

A simple PID controller can be used to keep track of these forces.

4.4. Desired finger-tip forces

Using equations (2), (3), (10), (12) and (13), the components of the ultimately desired actuator force are known as

$$\begin{bmatrix} F_{mi}\cos\phi_i \\ F_{mi}\sin\phi_i \end{bmatrix} = \boldsymbol{f}_{i,hold} + \boldsymbol{f}_{i,grasp} + \boldsymbol{f}_{i,motion} + \begin{bmatrix} F_{si}\sin\phi_i \\ -F_{si}\cos\phi_i \end{bmatrix}. \tag{14}$$

Solving these equations will give the magnitude of the actuator (motor) force F_{mi}. Though it seems strange how the direction of this force, i.e. ϕ_i, will be determined, as the ME joint determining this parameter is passive and has no actuators with it. This is true when an ME is considered completely independent and having no interactions with the other agents of the system. However, once, a finger-tip is in contact with the object and the object in contact with other agents of the system, it becomes a closed loop system and many constraints are developed. As all these constraints are directly related to the active and passive forces generated by all the agents, hence a balance in these forces, as guaranteed by equation (10), will guarantee a correct ϕ_i also as desired for a certain manipulation element.

4.5. Interpretation of an event

In this paper, a point-to-point task is interpreted as an event. For this purpose, a given manipulation task is divided into many small PTP sub-tasks, as shown in Figure 7. For an event-based controller, to be discussed next, this PTP task becomes a main reference command. The controller is supposed to develop the forces which can generate motions in an ME towards minimizing the error in the reference event command and the actual robot status irrespective of whatever amount of time is spent for this job.

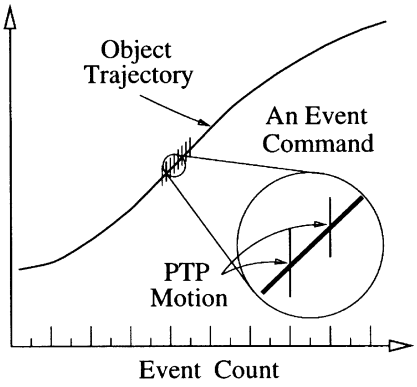

Fig. 7. Interpretation of an event.

4.6. The controller

Figure 8 shows the block diagram of the distributed event-based controller for the proposed scheme. The main functional blocks are explained below.

- **The Supervisor:** The proposed scheme utilizes a supervisory controller-like architecture. However, here, the supervisor does nothing but manages the communication of necessary data from and to the individual MEs. Moreover, it also computes the current object posture and internal forces, the information used by all agents in the system, and communicates this data back to the individual MEs. In this sense, it only acts as a **lookup table** for the robots from where they can access the control parameters and can write back the results of their own executions. The main task parameters are assigned to this supervisor. The supervisor divides this task into event commands. It, then, accesses the current task and robots' status and again distributes them among all the robots along with the assigned event parameters. However, if some force sensors are available at the finger-tips of the MEs, the amount of data to be communicated becomes very small. Again, at the end of an event execution, the supervisor gathers current data from all the robots and estimates the status of the task and the robots.
- **Finger-Tip Trajectory Generation:** This block of an individual robot controller accesses the object posture and reference manipulation commands from the supervisor and computes the reference finger-tip trajectories using equation (9).
- **Trajectory Modification for Internal Forces:** This block modifies the desired finger-tip trajectories for development of proper grasping forces using the illustrations of Figure 6, as defined in equation (12), if there arises a need of it. Normally, this block is needed only in the start of task execution when forces are not known exactly. Moreover, it is also needed when the internal forces are to be controlled during the object manipulation. This block computes ${}^{a}\boldsymbol{l}'_{fi}$ using equation (11), and again computes the modified trajectories using equation (9), finally.
- **Finger-Tip Position:** This block receives data from the sensors, i.e. the length of the push rod l_i and the angle of rotation of the robot's base θ_i, and computes the actual current finger-tip position using the kinematics of the robot.
- **Controller:** This is an event-based controller which generates the forces for motion, using equation (13). At the output of this block, a combination of forces for motion and grasping is available, as it accepts an already modified reference trajectories set.
- **Finger Force Computation:** If a force sensor is not available at a finger-tip, equation (8) is used to compute the resultant opposing forces at this finger-tip due to all other MEs. The data for the available finger-tip forces is acquired from the lookup table. Though, it is not a fresh data but the one in the last event, yet it works well as an event defines a very small motion. However, if the actual sensors are available, the real forces can be used.
- **Gain:** This block converts the desired finger-tip force components of equation (14), computed at the node preceding it, to a resultant motor force command. This command is processed with factors dependent on motor/actuator parameters and consequently a motor torque command is obtained.

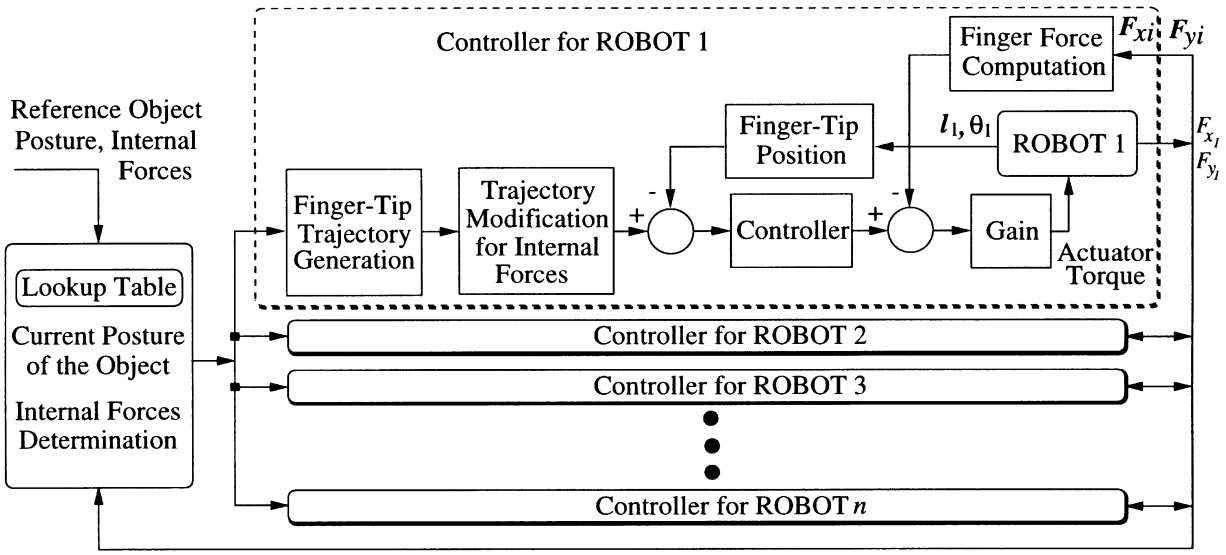

Fig. 8. Block diagram of the controller.

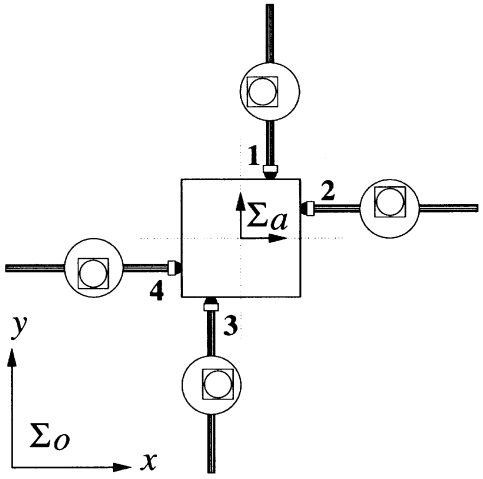

Fig. 9. System configuration used in simulations and experiments.

The components inside the dashed enclosure define an individual robot's controller. All the robots (MEs) realize a same controller and then all these controllers are supervised by the lookup table.

Each controller owns a **communication/event tag**, e.g. the event count, which tells the other controllers and the supervisor that it has completed its job for the current event and is ready to communicate the results and access new event parameters. This tag is incremented at the completion of each event, i.e. a PTP motion, as commanded by the interpolated object trajectory.

5. MULTI-TASK SIMULATIONS

The proposed distributed event-based control scheme is simulated for a system with four MEs handling a square object. The initial configuration of the system and the arrangement of MEs is shown in Figure 9. This type of grasping configuration has been found very efficient for perfect force closure definition on an object for its immobility, and for all possible manipulations in a plane.[14]

A total of five control programs are necessary, one for the supervisor and one for each robot. MATLAB™ on a multiuser workstation has been used to run the five simulation programs in collaboration.

Figures 10 and 11 show the simulation results for two different object manipulations. The dimensions are in *meters* for x and y positions and in *radians* for orientation. Figure 10 shows the results when the object is moved in the $x-y$ plane from $(-0.04,\ 0.04,\ 0.00)$ to $(0.04,\ -0.04,\ 0.00)$, while Figure 11 depicts the results for both translation and rotation from $(-0.04,\ 0.04,\ 0.10)$ to $(0.04,\ -0.04,\ -0.10)$. The commanded parameters correspond to ${}^{o}x_a$, ${}^{o}y_a$ and ψ_a, respectively. The plots in each set show the results for the object reference and current trajectories, individual fingers' reference and current trajectories, and the variations in the magnitude of internal forces on the object F_r, which is one-dimensional in this case. The internal force has not been controlled and hence there are variations due to changes in the posture of the object.

An inspection of the simulation results reveals that the MEs have been able to manipulate the object correctly, though there are little errors which can be neglected. The main reason for these errors can be thought to be an improper selection of control gains. However, the overall impression of the simulation results is quite encouraging and hints that the introduced control method should work well for the practical systems too.

6. EXPERIMENTAL IMPLEMENTATION AND DISCUSSION

The proposed control scheme has been experimentally implemented for the same system shown in Figure 9. For this experimental procedure, the testbed contains a UNIX workstation which acts as a "*host*", while the robots are directly controlled by a VME-based embedded controller

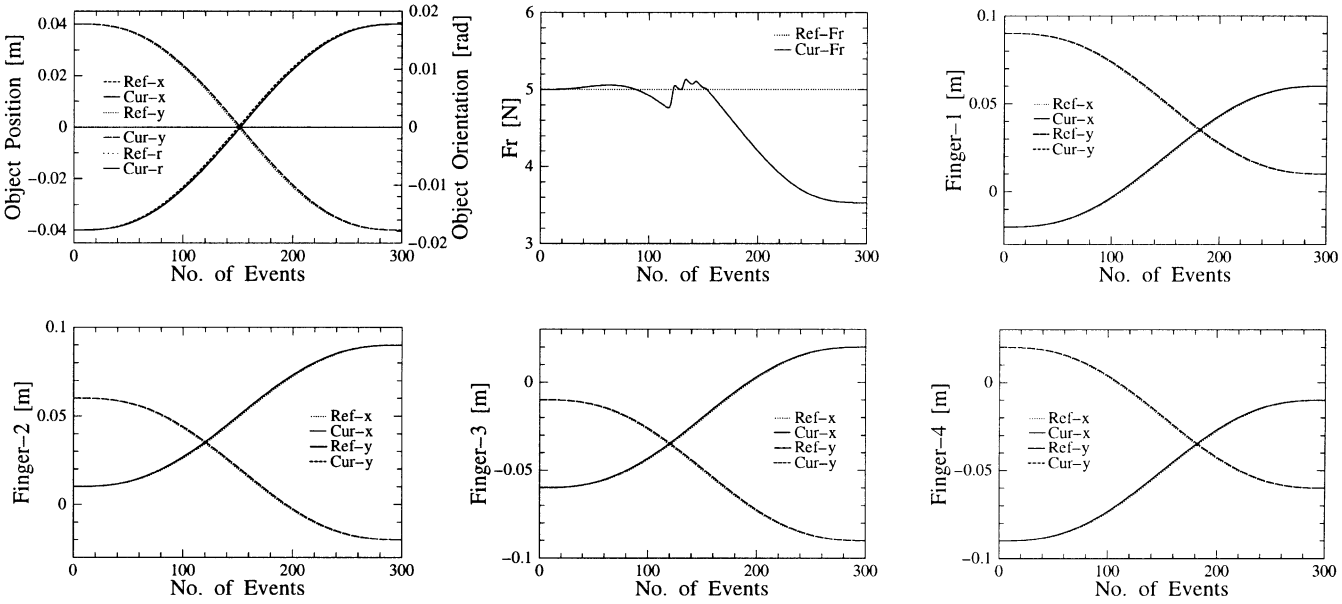

Fig. 10. Object manipulation in x–y plane.

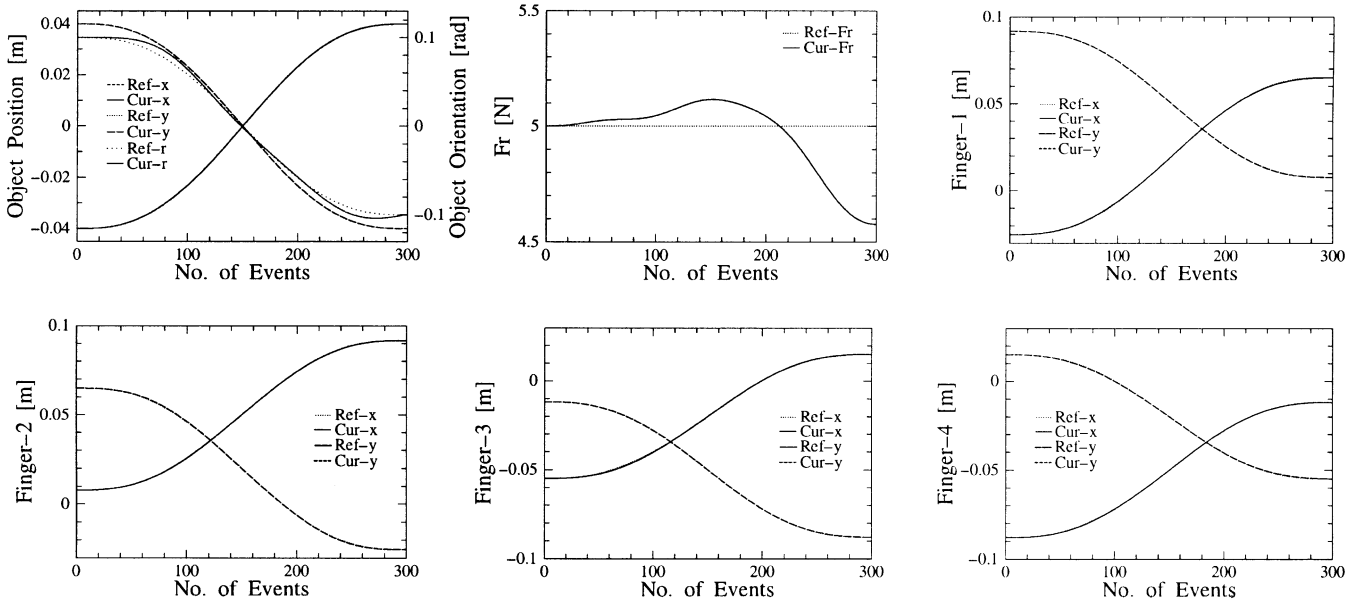

Fig. 11. Object manipulation in x–y plane and rotation.

operating with VxWorks™ as its real-time OS, regarded as "*target*". The lookup table runs on the host while the individual programs run on the target as its different tasks. The data communication is done through files as time is not of too much concern. However for better performance, the lookup table can also be run as a separate task on the same target or the data communication between the host and the target can be realized with some other efficient communication protocol. However, as the purpose here has been to verify the proposed control method, hence little attention is paid towards this aspect.

Figures 12 and 13, again, show the results of two different task executions in the same order as that for the simulation results. Figure 12 shows the results when the object is manipulated in the $x-y$ plane from (0.00, 0.00, 0.00) to (0.02, -0.02, 0.00), while Figure 13 reports the results when the object is commanded for an orientation change also, along with an $x-y$ motion, from (0.00, 0.00, 0.00) to (0.02, -0.02, 0.20).

Both experiments have proved the validity and applicability of the proposed theory in this paper. Though there are small initial errors which are mainly due to the static friction in the push rods. This static friction exists at the start of each event though the effect is greater when the commanded step in an event is very small. Moreover, another factor is the improper modeling of the passive forces which have been linearized though they are not. While the selection of control gains, manipulation speed and initial magnitude of grasping forces also play a vital role. That is why, here a larger variation in the grasping forces can be observed compared to the simulation results where these considerations are not of much significance

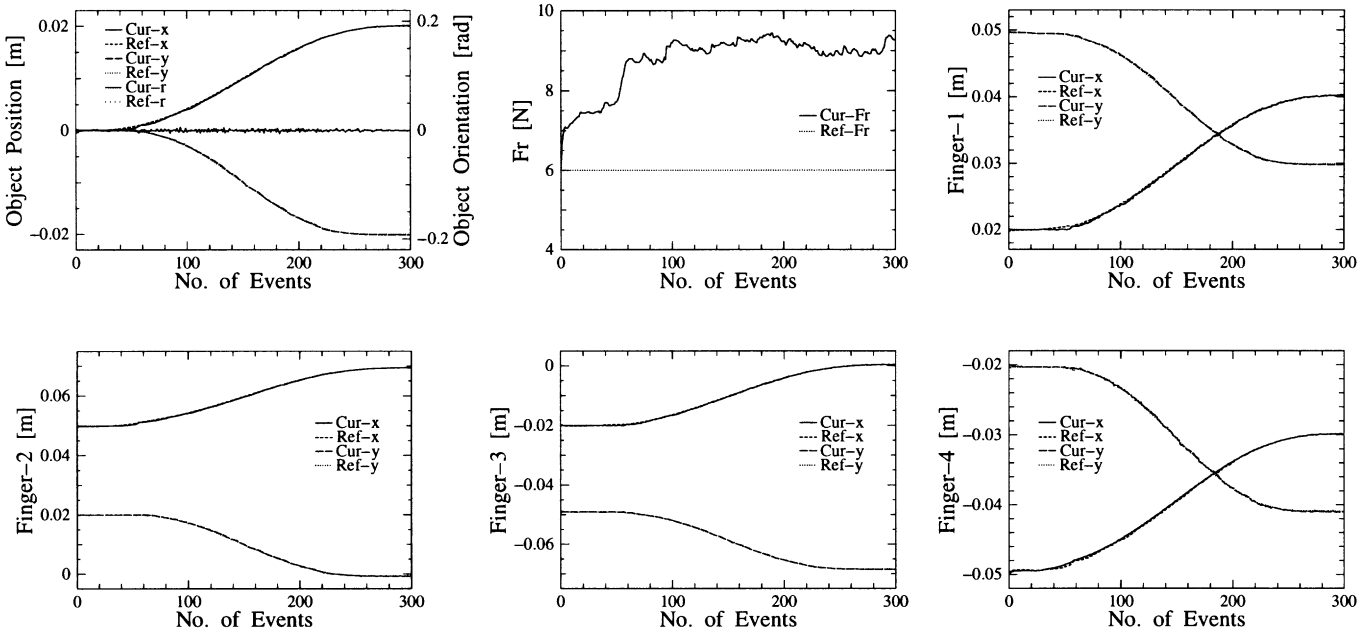

Fig. 12. Object manipulation in x–y plane.

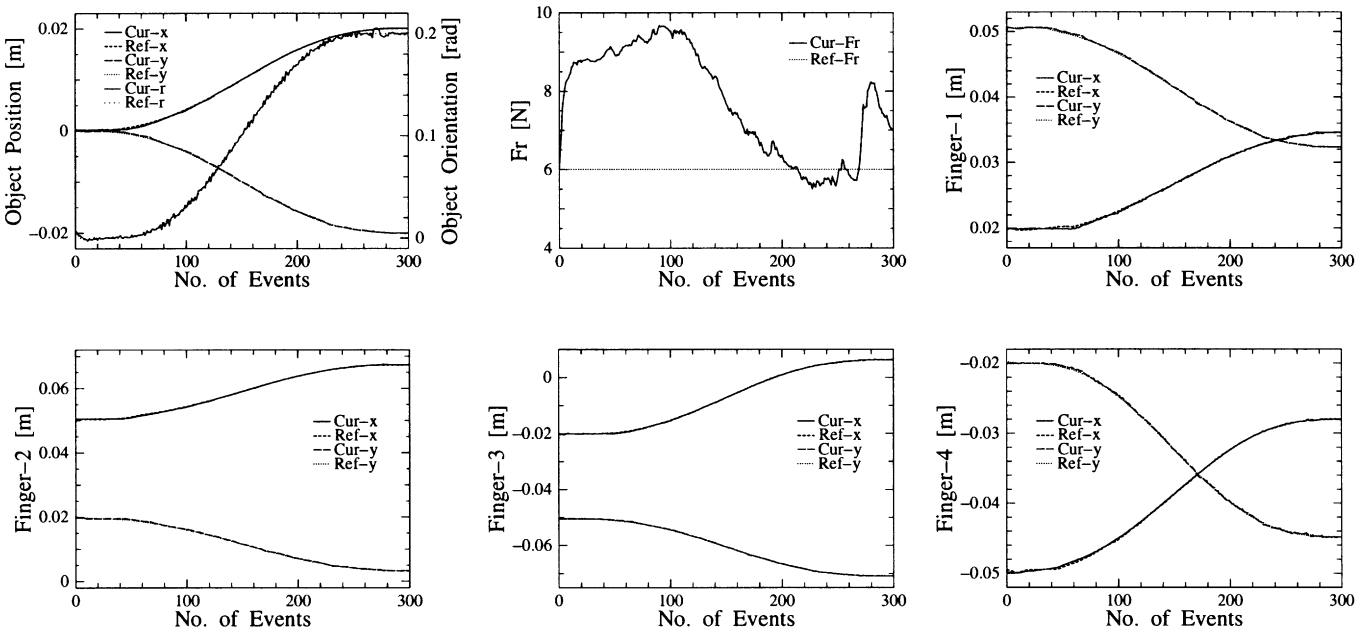

Fig. 13. Object manipulation in x–y plane and rotation.

as only the static model of the system is considered. However, in spite of all these factors, the system has been able to manipulate the object to the desired posture. Using these results, it can be concluded that even very simple systems can work in a distributed environment autonomously when they can interact and cooperate with the other agents in the same system. The proposed control method can be easily utilized for any general cooperative manipulation system with a few desired modifications.

7. CONCLUSIONS

Distributed control schemes have been used for very intelligent and autonomous systems only, especially for mobile robots. There are almost no examples for these control methods being used for non-autonomous and under-actuated systems. Moreover, the manipulation of an object, after establishing a cooperative grasp, is always realized with centralized controllers using sophisticated sensors.

This paper has given the distributed event-based control strategy for a non-autonomous, immobile and under-actuated multiple manipulator system using very few sensors. The development of actuator forces for each individual robot and their cooperation strategies are introduced. The development of grasping forces by trajectory reshaping is an easy concept and a fast method. Finally the simulations and experimental implementations have proved the applicability of the distributed and event-based control schemes, in the proposed way, for even very poor systems.

ACKNOWLEDGEMENTS

Acknowledgements are due to the Government of Pakistan, Ministry of Science and Technology, whose S&T Scholarship Program made it possible for the first author to conduct graduate studies and, consequently, produce this work. This work was partly supported by the Japan Society for the Promotion of Science (Research for the Future Program: Project No. 96P00802).

References

1. A.J. Koivo and G.A. Bekey, “Report of workshop on coordinated multiple robot manipulators: planning, control, and applications”, *IEEE J. Rob. and Automat.* **4**, No.1, 91–93 (1988).
2. B.R. Donald, J. Jennings and D. Rus, “Analyzing teams of cooperating mobile robots”, *Proc. of IEEE ICRA* (1994) **Vol. 3**, pp. 1896–1903.
3. N. Mitsumoto, T. Fukuda, F. Arai and H. Ishihara, “Control of the distributed autonomous robotic system based on the biologically inspired immunological architecture”, *Proc. of IEEE ICRA* (1997) **Vol. 3**, pp. 3551–3556.
4. H. Asama, T. Fukuda, T. Arai and I. Endo, *Distributed Autonomous Robotic Systems* (Springer-Verlag Tokyo, 1994).
5. Z.D. Wang, E. Nakano and T. Matsukawa, “Realizing cooperative object manipulation using multiple behavior-based robots”, *Proc. of IROS* (1996) **Vol. 1**, pp. 310–317.
6. M.N. Ahmadabadi and E. Nakano, “Constrain and move: A new concept to develop distributed transferring protocols”, *Proc. of IEEE ICRA* (1997) **Vol. 3**, pp. 2318–2325.
7. N. Xi, T.J. Tarn and A.K. Bejczy, “Intelligent planning and control for multirobot coordination: an event-based approach”, *IEEE Trans. on Rob. and Automat.* **12**, No. 3, 439–452 (1993).
8. K. Munawar and M. Uchiyama, “Distributed event-based control of unifunctional multiple manipulator system”, *Proc. of IEEE ICRA* (1998) **Vol. 2**, pp. 1817–1822.
9. K. Munawar and M. Uchiyama, “Experimental implementation of distributed event-based control of multiple unifunctional manipulators”, *Proc. of IEEE ICRA* (1999) **Vol. 2**, pp. 1213–1218.
10. K. Minami, S. Kawamura and M. Esashi, “Fabrication of Distributed Electrostatic Micro Actuator (DEMA)”, *IEEE J. Microelectromech. Sys.* **2**, No. 3, 121–127 (1993).

11. M. Yamaguchi, S. Kawamura, K. Minami and M. Esashi, "Control of distributed electrostatic microactuators", *J. Micromech. Microeng.* **3**, 90–95 (1993).
12. K. Minami, H. Morishita and M. Esashi "A bellows-shape electrostatic microactuator", *J. Sensors and Actuators* No. 72, 269–276 (1999).
13. K. Munawar and M. Uchiyama, "Slip compensated manipulation with cooperating multiple robots", *Proc. of 36th IEEE CDC* (1997) **Vol. 2**, pp. 1918–1923.
14. T. Yoshikawa, "Passive and active closures by constraining mechanisms", *Proc. of IEEE ICRA*. (1996) **Vol. 2**, pp. 1477–1484.